Attorney Docket No.: 019680-007800US
Client Reference No.: P000855

PATENT APPLICATION

UNIVERSAL RAID CLASS DRIVER

Inventor(s): Qiming Zhu, a citizen of China , residing at
44966 Winding Lane
Fremont, CA 94539

Assignee: NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA, 95050

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 415-576-0200

UNIVERSAL RAID CLASS DRIVER

BACKGROUND OF THE INVENTION

[0001] The present invention relates to the field of data storage devices. Computers often store large quantities of data, including data such as music, video, games, applications, and other valuable information, on hard disk drives and other data storage devices, which are referred to herein generally as "disks". As users' storage needs increase and the price of disks fall, computers often employ multiple disks to meet their storage needs.

[0002] To improve performance, reliability, and efficiency in using multiple disks for data storage, two or more disks can be combined into a single "logical" disk. One common disk architecture for combining multiple disks is RAID (Redundant Array of Independent Disks). Although RAID systems can provide improved disk performance and/or reliability, RAID systems are complicated to set up and configure. Additionally, RAID systems typically require specialized disk controllers and driver software. Thus, RAID systems are typically too expensive and too difficult for use by most computer users.

[0003] Additionally, because some current RAID systems require specialized disk controllers, they are limited in the number and type of disks that can be combined. For example, it is virtually impossible for current RAID systems to combine an EIDE disk and a SCSI disk in the same RAID system. In other RAID systems, the computer operating systems combines disk partitions, rather than the underlying disk itself, to create the RAID system. Because disk partitions are defined by the operating system, the disk partitions, and hence the entire RAID system, are inaccessible until the operating system has loaded. Thus in these implementations, the computer cannot be booted from the RAID system.

[0004] It is therefore desirable for a system and method to enable users to easily combine two or more disks into a bootable RAID system without specialized disk controllers. It is further desirable to be able to create RAID systems using disks of different types, controllers, and interfaces.

BRIEF SUMMARY OF THE INVENTION

[0005] The invention generally is a RAID class driver model that enables users to easily combine two or more disks into a bootable RAID system without specialized disk controllers and allows the creation of RAID systems using disks of different types, controllers, and interfaces. A RAID class driver is initialized in response to the identification of a RAID controller. Disk controllers return RAID-specific device identifications, rather than standard disk device identifications, for each disk to be included in the RAID system. The RAID class driver binds a RAID-specific functional interface to each disk having a RAID-specific device identification and combines the disks into a disk object representing the entire RAID system. The disk object provides the operating system with a standard disk device identification. The operating system loads a standard disk driver to interface with the disk object, thereby enabling transparent access to the RAID system.

[0006] In an embodiment, a storage disk device driver architecture comprises a RAID class driver having a physical device object representing a RAID system. The RAID system includes a plurality of disks. Each disk is associated with a functional device object adapted to interface with a physical device object representing the disk. The physical device object representing each disk provides a RAID-specific device identification. In a further embodiment, the physical device object that provides a RAID-specific device identification is included in a disk controller driver adapted to interface with a disk controller. In still another embodiment, the physical device object representing the RAID system is adapted to provide a standard disk device identification to an operating system.

[0007] In an embodiment, the RAID class driver is adapted to combine each disk into a RAID system. In one aspect of this embodiment, in response to receiving a request to write a data block to RAID system, the RAID class driver is adapted to mirror the data block on at least a portion of the plurality of disks via the associated functional device objects. In another aspect of this embodiment, in response to receiving a request to write a first and second data block to RAID system, the RAID class driver is adapted to write via the associated functional device objects the first data block to a first portion of the plurality of disks and to write via the associated functional device objects the second data block to a second portion of the plurality of disks. In another aspect of this embodiment, in response to receiving a request to write a first and second data block to RAID system, the RAID class driver is adapted to write via the associated functional device objects an error correction block to a portion of the plurality of disks.

[0008] In yet a further embodiment, a first portion of the plurality of disks is associated with a first disk controller of a first type and a second portion of the plurality of disks is associated with a second disk controller of a second type. In one aspect of this embodiment, the first type is an EIDE type controller and the second type is a SCSI type controller. In another aspect, the first type is a serial ATA type controller and the second type is a parallel ATA type controller. In a third aspect, the second type is a controller for an external disk.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The invention will be described with reference to the drawings, in which:

Figure 1 illustrates a computer system suitable for implementing an embodiment of the invention;

Figures 2A, 2B, and 2C illustrate example RAID system implementations that may be created by an embodiment of the invention;

Figure 3 illustrates an example prior device driver architecture enabling a computer to access disks;

Figure 4 illustrates a device driver architecture enabling a computer to access disks and RAID systems according to an embodiment of the invention; and

Figure 5 illustrates optimized RAID system data access enabled by an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0010] Figure 1 is a block diagram of a computer system 100, such as a personal computer, video game console, personal digital assistant, or other digital device, suitable for practicing an embodiment of the invention. Computer system 100 includes a central processing unit (CPU) 105 for running software applications and optionally an operating system. In an embodiment, CPU 105 is actually several separate central processing units operating in parallel.

[0011] CPU 105 is connected with Northbridge 110 via CPU bus 112. The Northbridge 110 passes data between the CPU 105 and the memory 115 via memory bus 117, and between CPU 105 and graphics processing subsystem 120 via graphics bus 122. Buses 112,

117, and 122 may be implemented as any type of data transport bus, including proprietary processor and memory buses, and AGP, PCI, PCI-X, and Hypertransport buses. In an alternate embodiment, CPU 105 includes a memory controller that interfaces with memory 115 directly, bypassing Northbridge 110.

[0012] The graphics subsystem 120 includes a graphics processing unit and optionally graphics memory for storing pixel data associated with output images. The graphics subsystem 120 periodically outputs pixel data for an image to a display device. Display device is any device capable of displaying visual information in response to a signal from the computer system 100, including CRT, LCD, plasma, and OLED displays. Computer system 100 can provide the display device with an analog or digital signal. In an alternate embodiment, the graphics processing subsystem is integrated with other computer system components, such the Northbridge 110.

[0013] Northbridge 110 is further interfaced with PCI bus 125. PCI bus 125 connects numerous peripheral devices with the computer system 100. Example peripheral devices include sound device 130, network interface 135, and disk controllers such as SCSI controller 140 and EIDE controller 145. Network interface 135 enables computer system 100 to communicate with other computer systems via an electronic communications network, and may include wired or wireless communication over local area networks and wide area networks such as the Internet.

[0014] The disk controllers enable access to non-volatile storage devices for applications and data and may include fixed disk drives, removable disk drives, flash memory devices, and CD-ROM, DVD-ROM, or other optical storage devices. In computer system 100, SCSI controller 140 is connected with one or more SCSI disks 185, and EIDE controller 145 is connected with one or more EIDE drives 180. EIDE drives may include drives using a parallel ATA interface (PATA) or a serial ATA interface (SATA). In a further embodiment, additional disks can be connected through additional disk controllers, which are typically used for internal drives, through USB and IEEE 1394 interfaces, which are typically used for external drives, and through wired and wireless network interfaces.

[0015] Southbridge 150 is also connected with PCI bus 125. Together, the Northbridge and Southbridge provide core logic functions of the computer system. Southbridge 150 enables I/O interfaces for numerous input and output devices, such as keyboards, mice, joysticks, touchpads, digital still and video cameras, printers, scanners, and digital music devices.

Southbridge 150 may support any number of I/O interfaces, such as USB 1.0 or 2.0 interface 160, IEEE 1394 interface 165, and other I/O interfaces 170, including serial, parallel, PS/2, and Bluetooth interfaces. Southbridge 150 also may support legacy peripheral devices through ISA bus 175. In a further embodiment, some or all of the peripheral devices 155 can be integrated into Southbridge 150, including disk controllers 140 and 145, sound device 130, network interface 135, and I/O interfaces 160, 165, and 170.

[0016] Figures 2A, 2B, and 2C illustrate example prior art RAID system implementations that may be created by an embodiment of the invention. As discussed above, a RAID system combines multiple disks into a single "logical" disk. RAID systems are typically divided into different categories or levels. There are numerous different levels, each of which includes some combination of mirroring, striping, and/or parity information. Each RAID level offers a different degree of improved reliability and/or higher performance. Discussed below are three of the more common RAID implementations; however, the invention is generally applicable to any combination of drives in any RAID implementation.

[0017] Figure 2A illustrates an example RAID 1 implementation 200, also referred to as mirroring, that provides improved reliability. As data is written to RAID system 200, RAID controller 205 writes the data in parallel to drives 210 and 215. In this RAID system, writing, or mirroring, data on drives 210 and 215 provides an automatic backup of data should one disk fail or become corrupted. In a further embodiment, a third disk 220 is connected with RAID controller 205 as a "hot spare." In the event of a failure of disk 210 or 215, RAID controller 205 will activate disk 220 and copy over the contents of the remaining operable drive. Subsequent data writes will then be mirrored on disk 220, rather than the defective drive.

[0018] Figure 2B illustrates an example RAID 0 implementation 235, also referred to as striping, that provides improved disk read and write performance. RAID 0 implementation 235 divides data into blocks and distributes the blocks over two disks, 245 and 250. RAID controller 240 reads and writes data from both drives in parallel, effectively doubling disk performance for disk accesses larger than one "block" of data. In further embodiment, data blocks are distributed over more than two drives, providing even greater performance.

[0019] Figure 2C illustrates an example RAID 5 implementation 270 that provides both improved disk performance and reliability. As with striping, RAID controller 275 divides data into blocks and distributes the blocks over two disks. Additionally, RAID controller 275

computes a parity or error correction block for every two data blocks. Each error correction block can be used by the RAID controller 275 to repair or reconstruct lost or corrupted data from its associated data blocks. As data in any block is changed, the RAID controller 275 rewrites not only the changed block but also updates the corresponding parity. In RAID 5 implementation, the parity blocks are alternately written to each of the disks to evenly distribute the load on disks. For example, RAID controller writes blocks 0 and 1 to disks 280 and 285 and the corresponding parity block to disk 290, and writes blocks 2 and 3 to disks 280 and 290 and the corresponding parity block to disk 285.

[0020] Figure 3 illustrates an example prior device driver architecture 300 enabling a computer to access disks. Microsoft® Windows® is one operating system that uses this device driver architecture for accessing disks. In device driver architecture 300, devices are located and accessed via a tree data structure supervised by a PnPManager driver. The PnPManager driver creates the device driver tree by starting at a root system node and enumerating a first level of connected "child" devices. A driver is loaded for each connected child device, which enables the enumeration of further child devices (e.g., the "grandchildren" of the root node). This is repeated for each level of child devices until all of the devices have been located and their respective drivers loaded.

[0021] For device driver architecture 300, the PnPManager will locate all of the controllers, such as a PCI controller and any disk controllers from the root node. The PnPManager creates a Physical Device object (PDO) for a disk controller 305. The disk controller PDO 305 has a specific device ID. From the device ID, the PnPManager determines the appropriate driver to be loaded. In the case of the disk controller PDO 305, a bus driver 310 is loaded. The bus driver 310 includes a disk bus Functional Device object (FDO) 315 for interfacing with the disk controller PDO 305.

[0022] The PnPManager enumerates the devices of the disk bus FDO 315 to locate Disk PDOs 320 and 325. Each disk PDO corresponds to a disk controlled by the disk controller. Like the disk controller PDO 305, disk PDOs 320 and 325 have their own device IDs. In the case of disk PDOs 320 and 325, the device ID specifies a generic disk device, "GenDisk."

[0023] In response to the device ID of "GenDisk," the PnPManager loads "Disk.Sys" drivers 327 and 329. Disk.Sys driver instance 327 includes an instance 330 of Disk Class FDO 330 for interfacing with disk PDO 320. The PnPManager enumerates the devices of disk class FDO to locate disk partitions PDOs 335 and 340. Disk partition PDOs 335 and

340 correspond to disk partitions on the disk controlled by Disk.Sys instance 327. From partition PDO 335, the PnPManager identifies the file system 345 for the disk partition, enabling the computer system to access data stored on the partition.

[0024] Figure 4 illustrates a device driver architecture 400 enabling a computer to access disks and RAID systems according to an embodiment of the invention. As in device driver architecture 300, a PnPManager or other operating system component responsible for managing and configuring devices scans a root node to locate and identify disk controllers. PnPManager creates disk controller PDOs for each disk controller identified. Disk controllers can be any hardware interface for one or more disks, including EIDE, SCSI, USB, and IEEE 1394 interfaces. For example, device driver architecture 400 includes disk controller PDOs 403 and 405.

[0025] The PnPManager determines the type of disk controller from a device ID provided by the disk controller PDOs, and loads the appropriate bus drivers, such as bus drivers 407 and 409. Each bus driver includes a Disk Bus FDO, such as Disk Bus FDOs 411 and 414, for interfacing with the corresponding disk controller PDO.

[0026] The PnPManager enumerates the disks connected with each disk controller using the corresponding Disk Bus FDO. A disk PDO is created for each disk connected with a disk controller. For example, disk PDOs 413 and 415 are created for two disks connected with the disk controller associated with bus driver 407. For disks that are part of a RAID system, the disk PDO does not have a device ID of "GenDisk." Instead, disks that are part of a RAID system have a different device ID, such as RAIDDisk, indicating to the PnPManager that a RAID class driver 417 should be loaded.

[0027] RAID class driver 417 is loaded if any disks located by a disk bus FDO have a device ID of "RAIDDisk" or any other RAID-specific device ID. RAID class driver 417 will create RAID disk FDO for each disk having a RAIDDisk device ID. For example, RAID disk FDOs 419 and 421 are created for the disk PDOs 413 and 415, respectively.

[0028] The purpose of the RAID class driver 417 is to combine two or more disks into a RAID system. In an embodiment, a user can designate disks to be combined in a RAID system using the computer system BIOS configuration utility. The user can also set the type or level of RAID system to be created. The RAID configuration settings, including the RAID level and the drives belonging to the RAID system, are stored in the computer system CMOS configuration memory, along with other BIOS configuration settings.

[0029] In an embodiment, the disk controllers, such as disk controllers associated with disk controller PDOs 403 or 405, read the CMOS configuration to determine if any or all of its connected disks are to be part of the RAID system, and if so, the disk controller will report the device ID of the disks as "RAIDDisk" rather than "GenDisk."

[0030] In an embodiment, RAID controller PDO 423 is also created from scanning the root node. The RAID controller PDO 423 has a device ID, such as RAIDBus, also associated with the RAID class driver 417. This ensures that the RAID class driver 417 is always loaded as a bus driver, and that an array PDO 425, discussed below, is created. In one embodiment, the RAID controller PDO 423 corresponds to a "phantom" controller that only has the responsibility for ensuring the RAID class driver 417 and array PDO 425 are loaded. In an alternate embodiment, RAID controller PDO 423 is associated with RAID controller hardware for performing one or more functions of the RAID system, such as automatically computing the parity or error correction block for two or more data blocks.

[0031] RAID class driver 417 creates an array PDO 425 representing the combination of all of the drives in the RAID system. If there are multiple independent RAID systems in a computer system, then RAID class driver 417 creates an array PDO for each RAID system. Array PDO 425 is created by enumerating the devices of the RAID bus FDO 427, which in turn is created to interface with RAID controller PDO 423. The array PDO 425 coordinates all data access to the disks forming the RAID system.

[0032] To interface with the operating system, the array PDO 425 returns a device ID of "GenDisk," which is the same as for any other disk. Thus, the operating system views the RAID system as a single ordinary disk, rather than a combination of disks. In response to the array PDO 425, the PnPManager loads a disk device driver 429, such as Disk.Sys. As with other GenDisk type disks discussed above, the disk device driver 429 includes a disk class FDO for interfacing with the array PDO 425 and one or more partition PDOs defining file systems accessible to the operating system.

[0033] By using a RAID class driver as a middle tier between the disk controller drivers and the operating system disk drivers, device architecture 400 enables RAID systems to be created and managed regardless of the degree of operating system support. Furthermore, the RAID class driver 417 can aggregate disks from many different controllers into a RAID system. Any type of disk controller can contribute disks to the RAID system; the only requirement is that the disk controller returns a RAIDDisk device ID for its associated disks.

[0034] Additionally, the RAID class driver can optimize data accesses with the RAID system. Figure 5 illustrates optimized RAID system data access 500 enabled by an embodiment of the invention. An IO Request Package (IRP) 505 is sent to the RAID class driver 510 to be written to the RAID system. In this example, IRP 505 includes four contiguous blocks of data to be written to the RAID system. Although these blocks of data are logically contiguous, the blocks may be stored in contiguous or non-contiguous blocks of system memory. For example, Block 0 is at address F0, Block 1 is at E0, Block 2 is at C0, and Block 3 is at D0. In an embodiment, each block can be accessed in system memory via a physical memory address or a virtual memory address. RAID class driver 510 receives the IRP 505 from the operating system and in this example determines that Blocks 0 and 2 should be written to disk 530 and Blocks 1 and 3 should be written to disk 540.

[0035] To optimize this operation, the RAID class driver 510 recognizes that two blocks need to be written to each disk. Rather than write each block in a separate disk operation, RAID class driver 510 groups the data accesses together into one data access for each disk. In an embodiment, RAID class driver 510 writes data to each disk by initiating a direct memory access (DMA) operation between system memory and the appropriate disk controller. In this embodiment, RAID class driver initiates a first DMA operation 515 transferring Blocks 0 and 2 to disk controller 525. RAID class driver 510 also initiates a second DMA operation 520 transferring Blocks 1 and 3 to disk controller 535. The disk controllers 525 and 535 receive the appropriate data blocks from the DMA transfers and write the data blocks to their respective disks.

[0036] This invention enables users to easily combine two or more disks into a bootable RAID system without specialized disk controllers and allows the creation of RAID systems using disks of different types, controllers, and interfaces. The invention additionally allows for further optimizations of disk access. Although the invention has been discussed with respect to specific examples and embodiments thereof, these are merely illustrative, and not restrictive, of the invention. Thus, the scope of the invention is to be determined solely by the claims.